LA-UR-14-25493
Approved for public release; distribution is unlimited.

Title: Optimal Kinodynamic Motion Planning in Environments with Unexpected Obstacles

Author(s): Boardman, Beth Leigh
Harden, Troy Anthony
Martinez, Sonia

Intended for: LANL Student Symposium

Issued: 2014-07-18

Optimal Kinodynamic Motion Planning in Environments with Unexpected Obstacles

Beth Boardman,[1,2] Troy Harden,[2] and Sonia Martínez[1]

[1]Department of Mechanical and Aerospace Engineering
University of California, San Diego

[2]Applied Engineering and Technology, Group 5
Los Alamos National Laboratory

July 16, 2014

Engineering Institute

UCSD Jacobs School of Engineering   Los Alamos NATIONAL LABORATORY EST. 1943

Overview

Outline

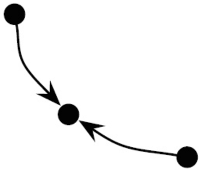
Motivation

7 Degree-of-freedom manipulator in a glovebox working with humans and/or other manipulators

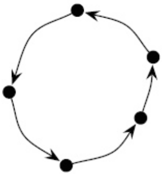
- Real-time path planning
- Want optimal (near-optimal) path
- Environment has moving/changing obstacles
- Replan, or “fix,” path quickly when obstructed by unexpected obstacles

A Short Introduction to Graph Theory [Bullo, 2009]

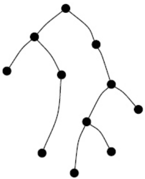
- A *graph* is a set of *vertices* (*nodes*) and *edges* (order pair of vertices)
- Graphs can be *undirected* or *directed* (i.e. can travel forward but not backwards)
- The edges can be *unweighted* or *weighted* (i.e. cost to get from one node to another)



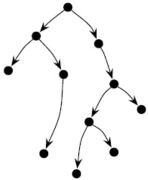
One Sink and Two Sources



Cycle



Undirected Tree



Directed Tree

Useful Notation

- *Configuration*, x: Complete specification of every position of the robot (i.e. position, velocity, orientation)
- *Configuration Space*, X: Set of all configurations
- *Obstacle Space*, X_{obs}: Set of all configurations that will cause a collision with itself or an obstacle
- *Free Space*, X_{free}: $X_{\text{free}} = X \setminus X_{\text{obs}}$
- ∂S: Boundary of a set S
- $\mathscr{O}$: New obstacle information (i.e. $\mathscr{O} \not\subset X_{\text{obs}}$)
- $\mathscr{T}_G$: Tree rooted at x_G
- $\mathscr{T}_I$: Tree rooted at $x_{I'}$

What is Motion Planning? [Lavalle, 2006]

- Determining how a robot should move to complete a given task
- Types of motion planners
  - Discrete Planners (i.e. D*, D* Lite)
  - Bug Algorithms
  - Sampling-Based Planners
    - Static Environments (i.e. RRT, RRT*, PRM, PRM*, RRT$^{\#}$)
    - Dynamic Environments (i.e. DRRT, RRF, LRF, ERRT, MP-RRT)

Details of the Rapidly Exploring Random Tree* [Karaman, 2011]

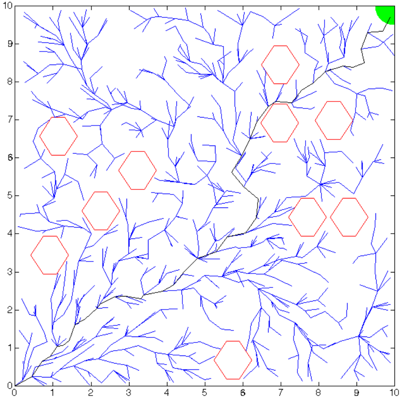


Figure: RRT* after 999 iterations.

Details of the Rapidly Exploring Random Tree* [Karaman, 2011]

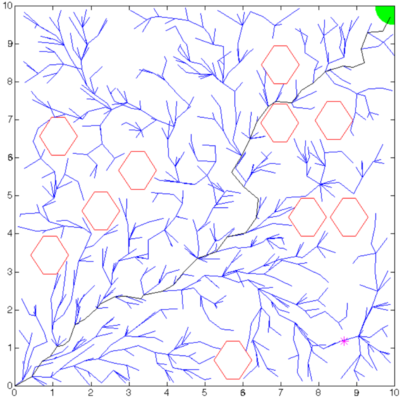


Figure: Randomly sample from the configuration space. Let's zoom in!

Details of the Rapidly Exploring Random Tree* [Karaman, 2011]

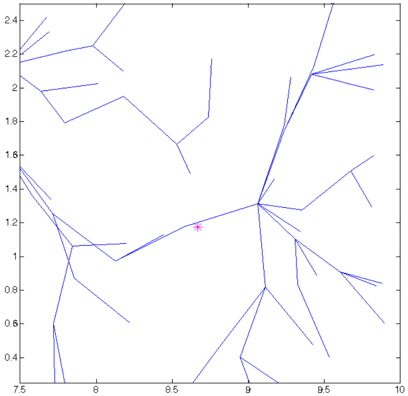


Figure: Zoomed in section with new sample, x_{new}, point to be added.

Details of the Rapidly Exploring Random Tree* [Karaman, 2011]

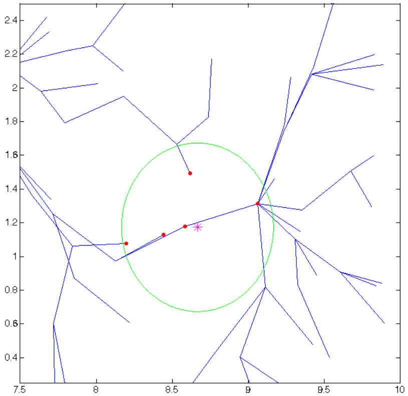


Figure: Find all neighbors within a given radius.

Details of the Rapidly Exploring Random Tree* [Karaman, 2011]

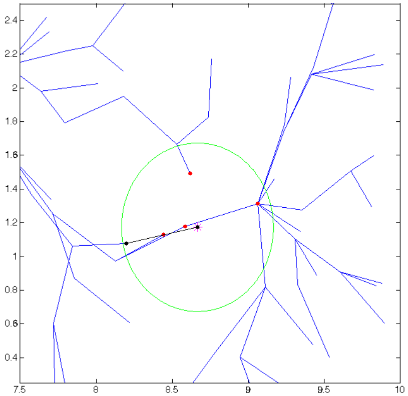


Figure: x_{new}'s parent: lowest cost-to-come and collision free edge.

Details of the Rapidly Exploring Random Tree* [Karaman, 2011]

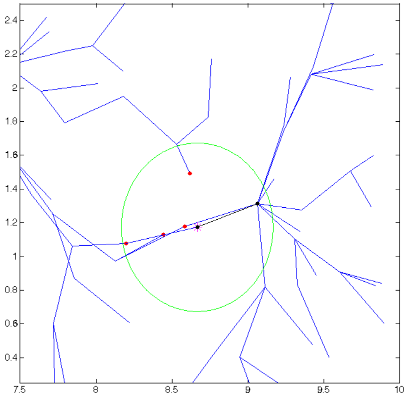


Figure: Rewire: Check neighbors as children of x_{new}.

Details of the Rapidly Exploring Random Tree* [Karaman, 2011]

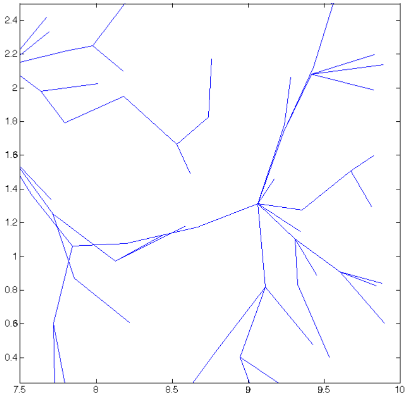


Figure: 1000 iterations complete!

Details of the Rapidly Exploring Random Tree* [Karaman, 2011]

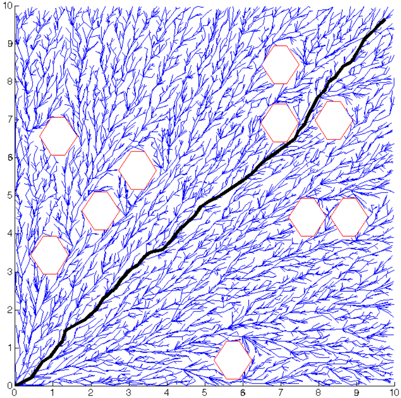


Figure: RRT* after 10,000 iterations

Outline

Goal Tree Algorithm Overview

Want to:

- Adapt the RRT* to handle unexpected obstacles, $\mathscr{O}$
- Reduce the runtime in comparison to rerunning the RRT*
- Recover the optimal path from $x_{I'}$ to x_G

Do this by:

- Build a tree, $\mathscr{T}_G$, whose branches all lead to the goal w.r.t X_{obs}
- Trim $\mathscr{T}_G$ to reflect conflicts with $\mathscr{O}$
- Build new RRT* in "affected area," $\mathscr{T}_I$
- Add branches from $\mathscr{T}_G$ to $\mathscr{T}_I$

Goal Tree Algorithm Details

- Robot is executing a path from $\mathscr{T}_G$
- $\mathscr{O}$ is found to be obstructing the path, $\mathscr{T}_G$ is trimmed
- Robot initializes a new tree, $\mathscr{T}_I$, at its current configuration
- $\mathscr{T}_I$ is extended toward a random sample
- An attempt is made to connect the new vertex in $\mathscr{T}_I$ to a vertex in $\mathscr{T}_G$
- If the attempt is successful, then the entire path from the vertex in $\mathscr{T}_G$ to x_G is added to $\mathscr{T}_I$

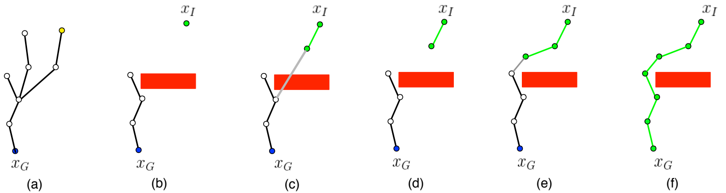
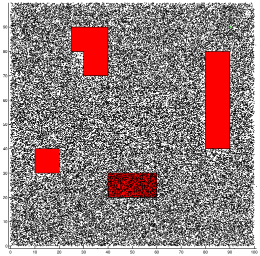


Figure: An illustrative example of how the Goal Tree algorithm works.

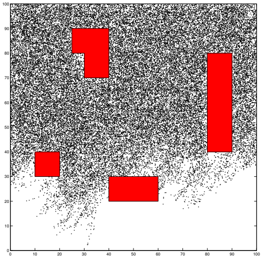
Outline

Optimality Overview

- Where can $\mathscr{T}_I$ sample to recover the optimal path?
  - In“affected area”
  - Around $\mathscr{O}$
- How can we reduce the algorithm runtime
  - Sample in a small region, R, (i.e. $R \subsetneq X$)
  - Limiting the collision checks to only obstacles in R



Vertex positions of a typical $\mathscr{T}_G$ before $\mathscr{O}$ is found.



Vertex positions of a typical $\mathscr{T}_G$ after $\mathscr{O}$ is found.

Robot with No Differential Constraints: Theorem

Theorem

Let X be a d-dimensional C-space such that $d \in \mathbb{N}$ and $d \geq 2$. Let the initial obstacle space be X_{obs} and let $\mathscr{O} \not\subset X_{\text{obs}}$ be new obstacle information. For simplicity, assume that $\mathscr{O} \cap X_{\text{obs}} = \varnothing$. If

1. *X is the Euclidean metric space,*
2. *$\mathscr{O} \subset R \subset X$,*
3. *R is convex, and*
4. *$x_{l'} \in R$*

then the GT algorithm will converge to a globally optimal path, π, as $n \to \infty$ by constructing $\mathscr{T}_{l'}$ using R and the $\mathscr{O}$ information and employing $\mathscr{T}_G$ with the previous X_{obs}.

Robot with No Differential Constraints: Proof

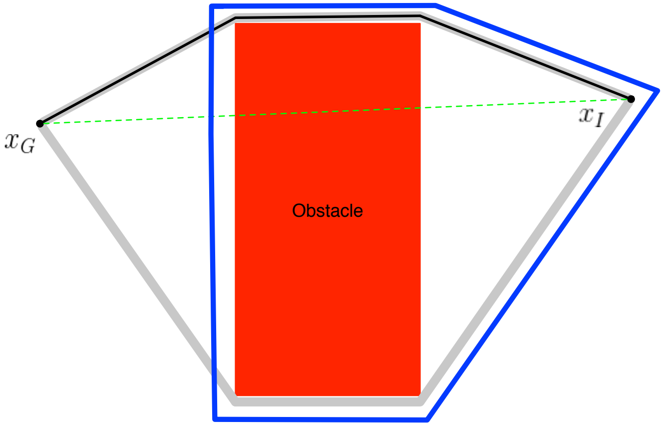


Figure: The gray lines represent possible paths to x_G with the optimal path in black. The boundary of a possible new sampling region is in blue. The region defined by the blue has all the properties specified in the previous theorem.

Robot with General Differential Constraints

Definition

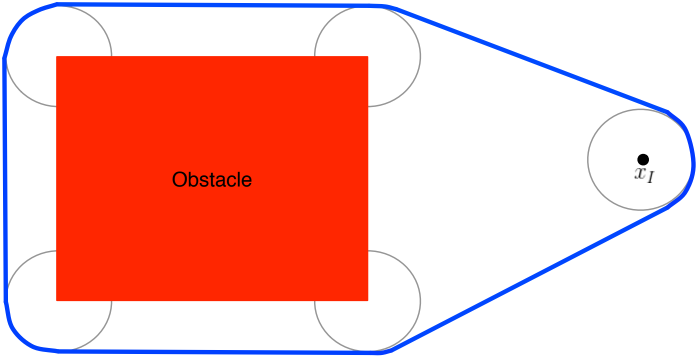
The shadow of x_G on $\mathscr{O}$, $\mathscr{S}_{\mathscr{O}}$, is the envelope or hull, as defined by position rather than configuration, formed by the geodesics from all configurations in X_{free} going to x_G that are in conflict with $\mathscr{O}$.

Theorem

Let $\mathscr{S}_{\mathscr{O}}$ be as in Definition 1. If the Goal Tree algorithm uses $\mathscr{S}_{\mathscr{O}}$ as the new sampling region to build $\mathscr{T}_I$, then it will converge to a globally optimal path as $n \to \infty$.

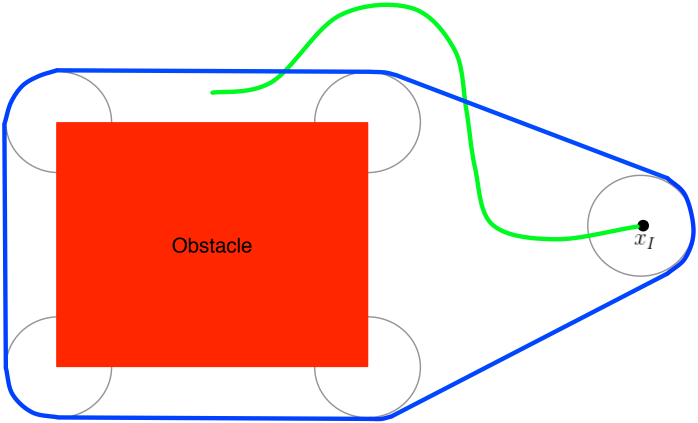
Sketch of Proof Can get from $x_{I'}$ to every outgoing configuration (configuration leaving R); from $\mathscr{T}_I$ construction. Every outgoing configuration has a path to x_G from $\mathscr{T}_G$; by the definition of $\mathscr{S}_{\mathscr{O}}$. $\square$

Dubins' Vehicle Robot



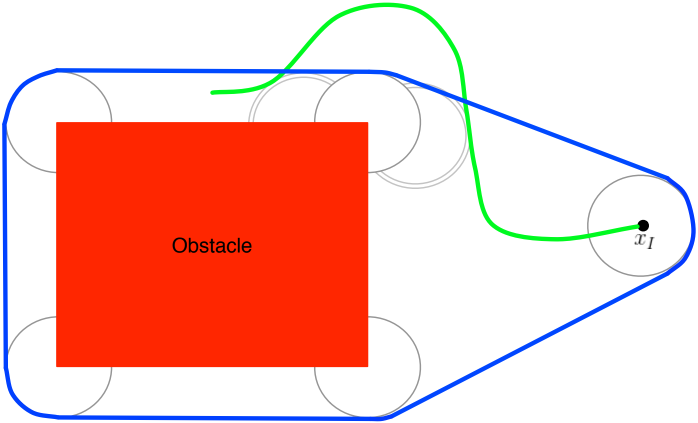
Possible new sampling region for a Dubins' vehicle (car with no reverse). Each circle has a radius of 2ρ (ρ is the minimum turning radius). The boundary of R is in blue.

Dubins' Vehicle Robot



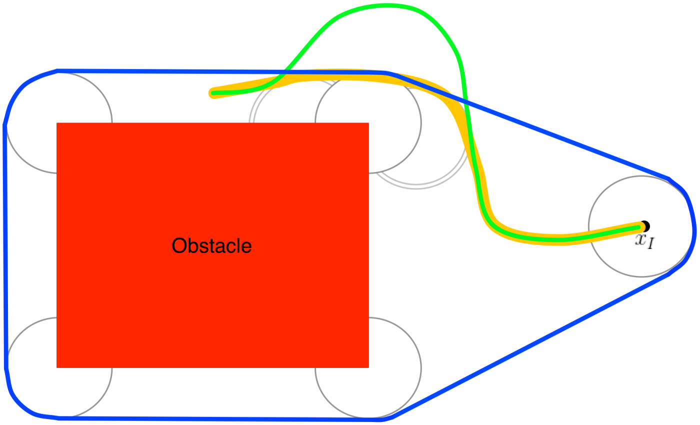
Let the green line be a trajectory from $x_{I'}$ to some other configuration inside R.

Dubins' Vehicle Robot



Take circles of radius ρ (or greater) and place them tangent to the path and ∂R as shown.

Dubins' Vehicle Robot



Create a new path that follows the old the circles, and ∂R. By construction this new path (yellow) has a shorter length than the original one (green).

Outline

Dubins' Vehicle Robot

- Configuration: x-position, y-position, and orientation θ
- Minimum turning radius, ρ
- Constant speed, v
- Control input, u

$$\dot{x}(t) = v\cos(\theta) \tag{1a}$$

$$\dot{y}(t) = v\sin(\theta) \tag{1b}$$

$$\dot{\theta}(t) = u, \quad |u| \leq \frac{v}{\rho}, \tag{1c}$$

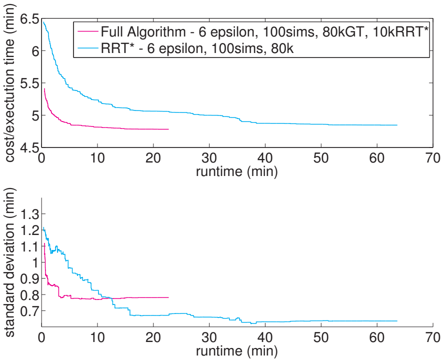


Figure: Minimum cost path to x_G as a function of algorithm runtime, averaged over 100 simulations, for a Dubins' vehicle.

Outline

Summary

- Introduced the Goal Tree algorithm for replanning in environments with unexpected obstacles
- The GT is asymptotically optimal
- A new sampling region R is proven to allow $\mathscr{T}_l$ to recover the optimal path
- Simulation results from the Dubins' vehicle show improved performance over the RRT*

Future Work

- Characterize R for higher dimensional systems with differential constraints
- How to determine $\mathscr{S}$ efficiently for use with the GT
- Extend the GT for use with multiple robots with multiple tasks
  - Root a tree at each task location
  - Each robot builds its own tree rooted at its current location
  - Possibly integrate cooperative control ideas

References

F. Bullo, J. Cortés, and S. Martínez (2009)
Distributed Control of Robotic Networks
Princeton University Press; Applied Mathematics Series
http://coordinationbook.info

S. LaValle (2006)
Planning Algorithms
Cambridge University Press
http://planning.cs.uiuc.edu

S. Karaman and E. Frazzoli (2011)
Sampling-Based Algorithms for Optimal Motion Planning
The International Journal of Robotics Research; Vol 30, No. 7, pgs 846–894